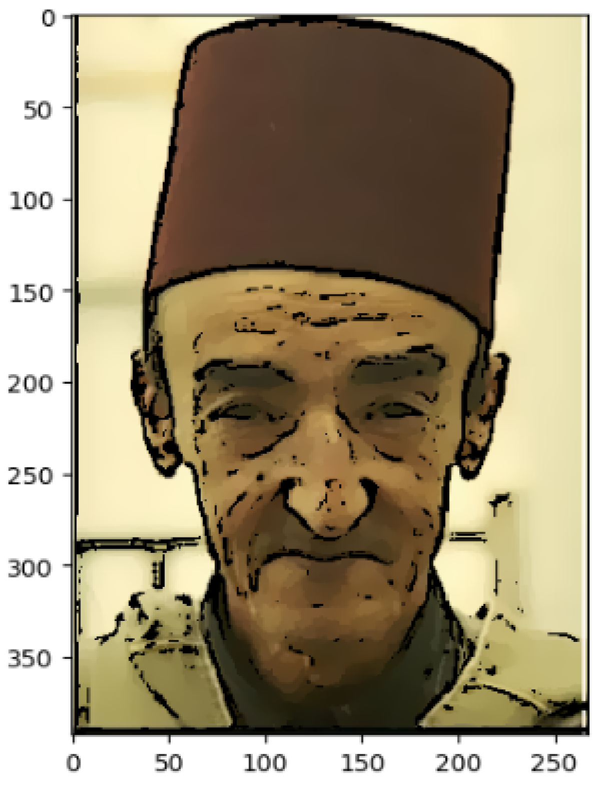
Assignment #1

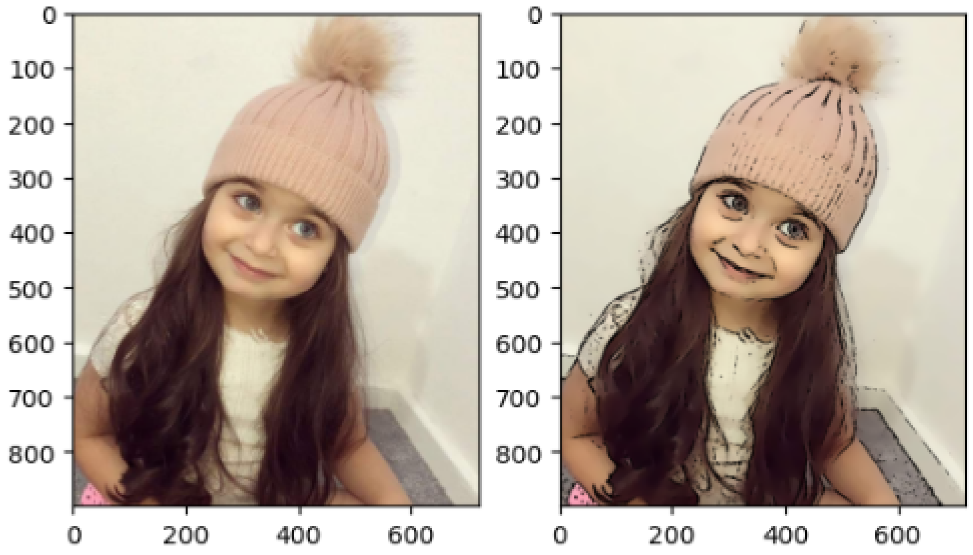
Computer Vision

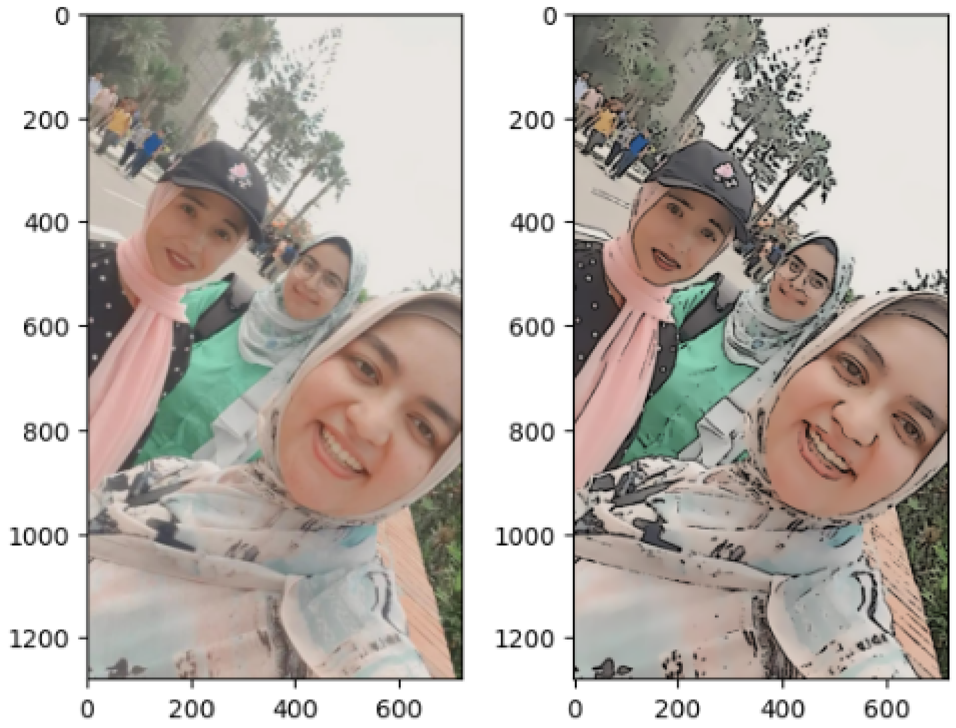
Name	id
Nada Mohamed Ibrahim	19016782
Rowan Nasser Edrees	19015686
Toka Ashraf Abo Elwafa	19015539

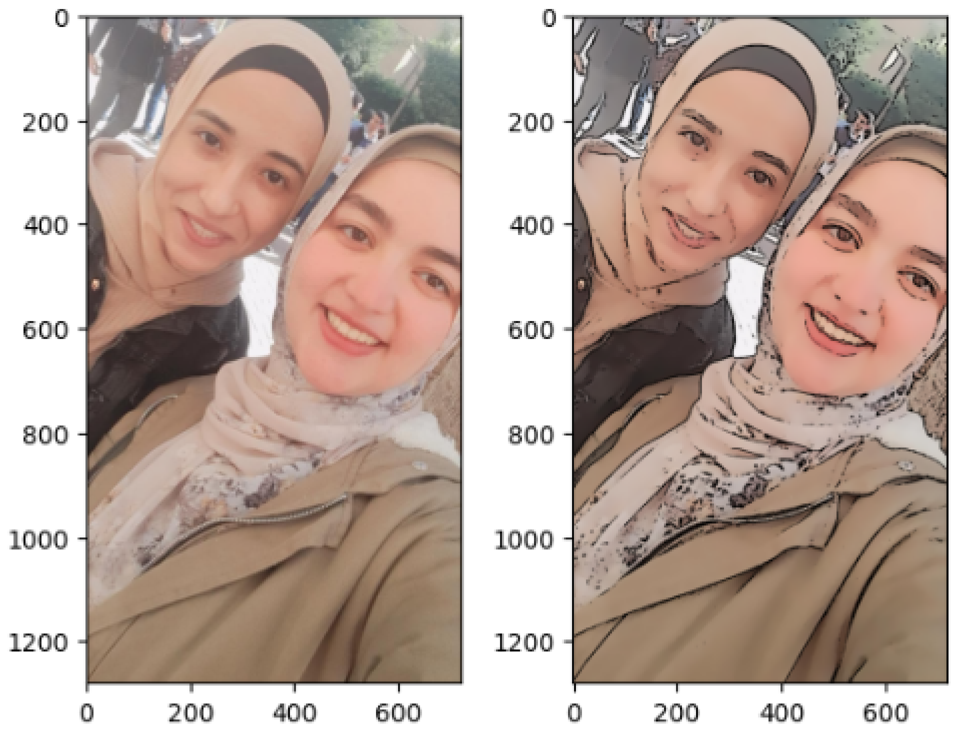
Part #1: Applying Image Processing Filters For Image Cartoonifyin

1. reading the path of image and displaying it in RGB color
2. converting the original image to a black and white image
3. using a median filter to reduce the noise in the image while keeping edges sharp before using a Laplacian edge filter.
4. using a Laplacian filter for edge detection (it produces edges with varying brightness)
5. applying a binary threshold to make the edges either white or black.
6. applying many small bilateral filters to smooth flat regions while keeping edges sharp
7. overlaying the edge mask (start with a black background and copy the painting pixels that aren't edges in the sketch mask)




0
100
200
300
400
500
600
700
800
0
200
400
600


0
200
400
600
800
1000
1200
0
200
400
600



[Part #2:Road Lane Detection Using Hough Transform:]()

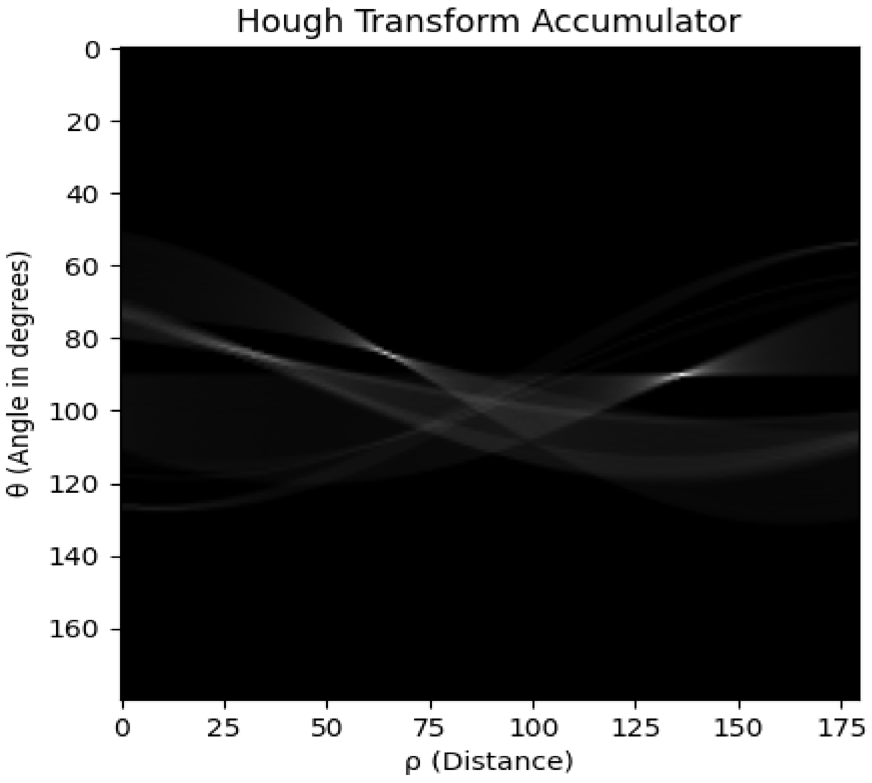
- **Conversion to Grayscale**
  The color image is converted to a grayscale image, simplifying subsequent processing.

- **Smoothing**
  Median blur is applied to the grayscale image to reduce noise and create a smoother version of the image.

- **Edge Detection**
  The Canny edge detection algorithm is used on the smoothed grayscale image to highlight significant changes in intensity, indicating edges and boundaries.

- **Extracting The Region Of Interest**
  It isolated a specific rectangular area that represented the top half of the image as a mask to be filled with white pixels then making AND operation between it and the image that resulted from appling canny algorithm on the smoothed image to get the region of interest.

- **Applying Hough Transformation**
  Firstly, since rho is the distance between the diagonal and the origin so diagonal should be computed using the width and the height of the masked image then determining the range of the angle theta to be from 0 to 180 and the distance rho to be from the negative of the diagonal to the diagonal with certain step then loop on the height and the width of the masked image to calculate the rho value from x, y and theta by the equation: x cos(theta) + y sin(theta) and get the rho index to vote by it and the theta index inside the accumulator array.
  The result accumulator array after drawn:



- **Applying The Non-Maximum Suppression**
  It executes after determining some indices of the accumulator that are greater than some value(e.g. 50) and then sorting these indices in descending order based on the corresponding values in the accumulator.
  it reduces the number of local maxima in the image. It ensures that only the most significant peaks are retained while suppressing less significant neighboring values based on threshold (e.g. 20).

- **Drawing The Lines**

  It is used to draw lines on the original image based on the results of the hough transform: rho values and theta values and the accumulator. It loops on the accumulator and takes the one that has votes greater than zero to calculate the endpoints of the line then determining the indices that will cut off the lines from it.

To give this result:

```
line_img = draw_lines(image, acc_after_suppression, rho_values, theta_values)
plt.imshow(line_img)
```

```
<matplotlib.image.AxesImage at 0x7959141a2290>
```